MA 3232 - Numerical Analysis
Exam I - Quarter II - AY 04-05

Instructions: Work all problems. Read the problems carefully. Show appropriate work, as partial credit will be given. One page ($8\frac{1}{2}$ by 11) of notes (both sides) and "Blue Books" permitted.

---

1. (30 points) a. Use two iterations of Newton's method to estimate the solution of

$$f(x) = x^3 - 6x + 3 = 0$$

between $x = 0$ and $x = 1$.

solution:

Newton's method uses the algorithm

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

which in this case reduces to

$$x_{n+1} = x_n - \frac{x_n^3 - 6x_n + 3}{3x_n^2 - 6}$$

Note that, also in this case, $f(0) = 3$ and $f(1) = -2$. Since x_0 is not specified, we must choose it.

Normally, the rule of thumb is to pick a value for x_0 that will produce the smallest possible initial residual. Therefore, since $f(1)$ is closer to zero, then, of the two given values, we should take $x_0 = 1$. In this example, however, because the residuals for both initial values are almost the same magnitude, then a very good argument can be made to pick as the starting value the midpoint, i.e. $x_0 = 0.5$. Probably the poorest choice, in general, would be $x_0 = 0$.

Proceeding from $x_0 = 1$, we have

$$x_1 = x_0 - \frac{x_0^3 - 6x_0 + 3}{3x_0^2 - 6} = 1 - \frac{(1)^3 - 6(1) + 3}{3(1)^2 - 6}$$

$$= 1 - \frac{(-2)}{(-3)} = \frac{1}{3} = 0.3333$$

solution:

Repeating the calculations produce the results shown in the table below:

n	x_n	$f(x_n)$	$f'(x_n)$	x_{n+1}
0	0.0000	−2.000	−3.000	0.3333
1	0.3333	1.037	−5.667	0.5163

Therefore $x_2 = 0.5163$. The table below summarizes the results you should have obtained with any of the possible starting values (including up thorough x_3, which will be needed for the next part):

x_0	x_1	x_2	x_3
0.0000	0.5000	0.5238	0.5240
0.5000	0.5238	0.5240	0.5240
1.0000	0.3333	0.5163	0.5240

b. Estimate the error in your answer to part a.

solution:

The standard method for error estimation in an iterative algorithm is to use the *next term*, which means conducting one more iteration, i.e.

n	x_n	$f(x_n)$	$f'(x_n)$	x_{n+1}
2	0.5163	0.03982	−5.200	0.5240

or $x_3 = 0.5240$. Therefore the error in $x_2 = 0.52163$ is approximately

$$x_3 - x_2 = 0.5240 - 0.5163 = 0.0077$$

(Note that, had you started with $x_0 = 0$, the approximate error would be 0.0002, and, if you had picked $x_0 = 0.5$, then the estimated error would be less than machine precision (i.e. < 0.0001). Interesting enough, that means that, in this case, starting with a slightly "worse" initial guess actually produces a correct result faster. Just goes to prove you don't necessarily win them all.)

c. Approximately how many iterations of the *bisection* method would have been required to achieve the same error?

solution:

In the bisection method, we know

$$e_n \leq \frac{L_0}{2^n}$$

where L_0 denotes the length of the original interval (which, in this case, assuming $x_0 = 0$ and $x_{-1} = 1$ is just one). Therefore, in this case

$$e_n \leq \frac{1}{2^n} \leq .0077 \qquad \Longrightarrow \qquad n \doteq 8$$

(Note that, had you started with $x_0 = 0$, and therefore obtained $x_2 = 0.5238$ with an approximate error of 0.0002, bisection would require about thirteen iterations to achieve the same accuracy. And, had you started with $x_0 = 0.5$, bisection would have required about fifteen.

d. Would the method of linear iteration, written as

$$x_{n+1} = \frac{x_n^3 + 3}{6}$$

have been a reasonable alternate method for solving this problem? (*Briefly* justify your answer.)

solution:

Probably **not**, although you need to be precise about what your criteria are!

Linear (fixed point) iteration will converge for this example, since

$$g(x) = \frac{x^3 + 3}{6} \qquad \Longrightarrow \qquad g'(x) = \frac{x^2}{2}$$

and therefore, for $0 < x < 1$, $|g'(x)| < 1$, which is sufficient to guarantee convergence. Therefore, the most reasonable criterion here is minimal function evaluation cost.

solution:

According to our result from part b., Newton's method appears to have converged to two digits after only two iterations. Since the number or accurate digits in a quadratically convergent method approximately doubles each iterations, our result would therefore almost certainly be accurate to four digits with one more iteration, and to eight or more digits with at most two more, i.e. a total of four iterations or eight function evaluations. (And really, in this case, if done properly, $f'(x)$ can be computed almost for free!) Linear iteration, which is only linearly convergent, would almost certainly converge more slowly here, especially as we increase the number of accurate digits required, although it might be comptetitive if we only require about four digits, since

$$g'(.5) = .125$$

So Newton is probably preferable here, especially since computing $f'(x_n)$ is very inexpensive!

2. (20 points) Consider the table of data:

x	e^{-x}
-0.15	1.16183
-0.10	1.10517
-0.05	1.05127
0.00	1.00000
0.05	0.95123
0.10	0.90484
0.15	0.86071

and the difference approximation

$$f'(x_n) = \frac{f_{n+1} - f_{n-1}}{2h} + \mathbf{O}\left(h^2\right)$$

(1) Approximate $f'(0)$ using step sizes of $h = 0.15,\ 0.10,$ and 0.05.

solution:

For step size $h = 0.15$, the approximation is

$$f'(0) \doteq \frac{f(.15) - f(-.15)}{2(.15)} = \frac{(0.86071) - (1.16183)}{.30}$$

$$= \frac{-0.301120}{.30} = -1.00373$$

Similarly, for $h = .10$:

$$f'(0) \doteq \frac{f(.10) - f(-.10)}{2(.10)} = \frac{(0.90484) - (1.10517)}{.20}$$

$$= \frac{-0.200330}{.20} = -1.00165$$

and for $h = .05$:

$$f'(0) \doteq \frac{f(.05) - f(-.05)}{2(.05)} = \frac{(0.95123) - (1.05127)}{.10}$$

$$= \frac{-0.100040}{.10} = -1.00040$$

(2) Is the behavior of the error in your approximations consistent with $\mathbf{O}\left(h^2\right)$ error? (Briefly *explain* your answer)

solution:

In this case we know the actual function, i.e. $f(x) = e^{-x}$ and therefore we know the correct answer:

$$f'(x) = -e^{-x} \qquad \Longrightarrow \qquad f'(0) = -1.0000$$

and can create the following table:

h	$Error$	$Error/h^2$
0.15	0.00373	0.166
0.10	0.00165	0.165
0.05	0.00040	0.160

Observe that, in each case, the error is almost exactly the same multiple of h^2, i.e.

$$Error \doteq 0.160h^2$$

This is exactly the behavior we expect from an $\mathbf{O}\left(h^2\right)$ error.

Equivalently, we can look at the ratios of the errors compared to the *squares* of ratios of the step sizes (since the method is supposedly $\mathbf{O}\left(h^2\right)$, and produce the following table:

h_1	$Error_1$	h_2	$Error_2$	$Error_1/Error_2$	$(h_1/h_2)^2$
0.15	0.00373	0.10	0.00165	2.261	2.25
0.10	0.00165	0.05	0.00040	4.125	4.00

This table confirms that reducing the step size by a given factor does in fact reduce the error by approximately the square of that factor, which is precisely the behavior we should expect from an $\mathbf{O}\left(h^2\right)$ method!

Alternatively, note that reducing the step size from $h = 0.15$ to $h = 0.05$, i.e. by a factor of three reduces the error by a factor of:

$$\frac{0.00373}{0.00040} \doteq 9.33$$

a result totally consistent with $\mathbf{O}\left(h^2\right)$.

3. (30 points) Consider the following table of data:

x_i	f_i	Δf_i	$\Delta^2 f_i$
0.0000	0.0000	0.5879	−0.1121
0.2000	0.5879	0.4758	−0.1468
0.4000	1.0637	0.3290	−0.1644
0.6000	1.3927	0.1646	−0.1644
0.8000	1.5573	0.0002	−0.1486
1.0000	1.5575	−0.1484	
1.2000	1.4091		

a. Using the most appropriate second-degree Newton-Gregory forward interpolating polynomial, approximate $f(0.63)$.

solution:

To uniquely determine a quadratic will require three data points. The closest three to $x = 0.63$ are:

$$x = .4000\ ,\quad .6000 \quad \text{and} \quad .8000 \qquad \Longrightarrow \qquad x_0 = .4000$$

The complete difference table, through the third forward differences (which will be needed for error estimates) is:

	x_i	f_i	Δf_i	$\Delta^2 f_i$	$\Delta^3 f_i$
	0.0000	0.0000	0.5879	−0.1121	−0.0347
	0.2000	0.5879	0.4758	−0.1468	−0.0176
$x_0 \rightarrow$	0.4000	1.0637	0.3290	−0.1644	0.0000
	0.6000	1.3927	0.1646	−0.1644	0.0158
	0.8000	1.5573	0.0002	−0.1486	
	1.0000	1.5575	−0.1484		
	1.2000	1.4091			

solution:

The second-degree Newton-Gregory polynomial, based on x_0 is defined as:

$$P_2(x) = f_0 + s\,\Delta f_0 + \frac{s(s-1)}{2}\Delta^2 f_0 \quad \text{where} \quad s = \frac{x - x_0}{h}$$

In this case,

$$s = \frac{0.63 - .40}{.20} = 1.15$$

and so

$$P_2(0.63) = (1.0637) + (1.15)(0.3290) + \frac{(1.15)(1.15-1)}{2}(-0.1644) = 1.4279$$

b. Estimate the error in your answer to part a.

solution:

The normal error estimate for the Newton-Gregory polynomial is the "next term," i.e. in this case

$$E_2(x) \doteq \frac{s(s-1)(s-2)}{3!}\,\Delta^3 f_0 = \frac{(1.15)(1.15-1)(1.15-2)}{6}(0.0000)$$

$$= 0.0000 \quad ????$$

c. Do you feel the error estimate you obtained in part b. is reasonable? (*Briefly* explain your answer!)

solution:

The "next term" error estimation rule is based on the exact error formula

$$E_2(x) = \frac{x - x_0)(x - x_1)(x - x_2)}{3!} f^{(3)}(\xi)$$

where ξ is some (unknown) point in the interval containing x_0, x_2 and x. In most cases, i.e. when the values in the appropriate column of the difference table do **not** differ by an order or more of magnitude, then we can estimate

$$f^{(3)}(\xi) \doteq \frac{\Delta^3 f_0}{h^3}$$

In this case, however, the value of zero for $\Delta^3 f_0$ does not appear representative of the values in that column, and therefore an error estimate of zero is **not reasonable**. A better (although probably conservative) estimate would probably be, because we are unsure of even the sign

$$E_2(x) \doteq \pm \frac{(1.15)(1.15 - 1)(1.15 - 2)}{6}(0.02) = \pm 0.0011$$

(Note the exact function here is

$$f(x) = 2e^{-(x/4)} \sin(\pi x/2) \qquad \Longrightarrow \qquad f(0.63) = 1.4280$$

and therefore the actual error is $E_2 = .0001$, and so our conservative estimate is actually about an order of magnitude high. (Which is far better than an order of magnitude low!)

4. (20 points) Assume you are working with a three-digit, decimal based, chopping machine. In this machine, the algorithm for computing exponentials produces

$$\mathbf{exp}(-\mathbf{5/17}) = .750$$

a. What is the forward error of this computation?

solution:

By definition, the forward error is the difference between the result produce by the given algorithm in the given machine ($\tilde{f}(x)$) and the exact result. In this case,

$$f(x) = e^{-5/17} = 0.7451888170... \text{ and } \tilde{f}(x) = \mathbf{exp}(-\mathbf{5/17}) = .750$$

Therefore, the forward error is

$$f(x) - \tilde{f}(x) = 0.7451888170... - .750 = -0.00481...$$

b. What is the backward error of this computation?

solution:

By definition, the backward error is the difference between the value of x passed to the algorithm and the value of $\tilde{x}$ that would have had to be passed in order for the computed result to be exactly correct. In this case, that means $\tilde{x}$ satisfies

$$f(\tilde{x}) = e^{\tilde{x}} = 0.750 \qquad \Longrightarrow \qquad \tilde{x} = \ln(.750) = -0.2876820...$$

Since the exact value of $x = -(5/17)$ is

$$x = -0.2941176...$$

then the backward error is

$$x - \tilde{x} \doteq -0.2941176 - (-0.2876820) = -0.0064356$$

c. Based on your answers to parts a. and b. above, does this machine's algorithm for computing e^x appear reasonably satisfactory for $-1 < x < 0$?

solution:

Based on parts a. and b., above the relative forward and backward errors are, respectively

$$\frac{f(x) - \tilde{f}(x)}{f(x)} \doteq= \frac{(-0.00481)}{0.745} \doteq -0.006$$

and

$$\frac{x - \tilde{x}}{x} \doteq \frac{(-0.00644)}{(-0.294)} \doteq 0.022$$

In a three-digit, decimal, chopping machine, machine precision is

$$\epsilon_{machine} = \beta^{1-n} = 10^{-2} = 0.01$$

Therefore, the forward error is less than machine precision, and the backward error is only slightly larger than machine precision. This is quite satisfactory performance for any algorithm, because, in fact, the best you can expect out of any algorithm is *backward stability*, i.e. that for any x,

$$\left| \frac{x - \tilde{x}}{x} \right| \leq K \epsilon_{machine}$$

where K is a "small", i.e. close to unity, constant. That certainly appears to be the case here, since the relative backward error computed above is only about twice machine precision.